pcg

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/865,300	05/24/2001	Mohamed Ben-Romdhane	259/008	5178

7590 10/05/2004

PROCOPIO, CORY, HARGREAVES & SAVITCH, LLP
530 B STREET
SUITE 2100
SAN DEIGO, CA 92101-4469

EXAMINER	
INGBERG, TODD D	
ART UNIT	**PAPER NUMBER**
2124	

DATE MAILED: 10/05/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)	
	09/865,300	BEN-ROMDHANE ET AL.	
	Examiner	Art Unit	
	Todd Ingberg	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 October 2002*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-109* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-109* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *5/24/2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claims 1 – 109 have been examined.

Specification

1. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1 – 89 and 108 – 109 are rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. The Examiner has shown a way to overcome this rejection below.

Claim 1
An information model **executing on a computer and stored on a computer readable-medium** representing a software architecture, comprising: a plurality of language independent format objects, each language independent format object representing a discrete component, wherein the components are structurally related into an information model according to a software architecture.

Claim 6
A system **executing on a computer and stored on a computer readable-medium** for creating an information model representing an inherent software architecture derived from a body of source code, comprising: an information model generator having a parser and a composer, the parser configured to extract program fragments from a body of source code and create a plurality of language dependent format objects, the composer configured to convert language dependent format objects into language independent format objects, wherein each language independent format object represents a discrete component in an information model.

Claim 10
A method **executing on a computer and stored on a computer readable-medium** for creating an information model representing an inherent software architecture derived from a body of

source code, comprising: extracting program fragments from a body of source code; converting the program fragments to a language independent format; and creating a plurality of language independent format objects, wherein each language independent format object contains related program fragments from the body of source and represents a discrete component in an information model.

Claim 18
A method **executing on a computer and stored on a computer readable-medium** for creating an information model representing an inherent software architecture derived from a body of source code, comprising: parsing a body of source code to extract data dependencies, functional dependencies, and control flow indicators; creating a plurality of language dependent format objects, each language dependent format object comprising an abstract syntax tree representing related program fragments within the body of source code; and
converting each language dependent format object into a language independent format object, wherein each language independent format object represents a discrete component in an information model.

Claim 26
A method **executing on a computer and stored on a computer readable-medium** for creating an information model representing a software architecture, comprising:
creating a plurality of language independent format objects, wherein each language independent format object represents a discrete component in an information model.

Claim 33
A system **executing on a computer and stored on a computer readable-medium** for manipulating an information model representing a software architecture, comprising: an information model viewer configured to provide a visual presentation of the information model representing the software architecture; and a system architect configured to modify the software architecture.

Claim 49
A method **executing on a computer and stored on a computer readable-medium** for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: merging two or more components into a compound component.

Claim 51
A method **executing on a computer and stored on a computer readable-medium** for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: fragmenting a compound component into its constituent components.

Claim 53
A method **executing on a computer and stored on a computer readable-medium** for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: modifying the hierarchical structure between two or more components in an information model; merging two or more components into a compound component; and fragmenting a compound component into its constituent components.

Claim 68
A method **executing on a computer and stored on a computer readable-medium** for manipulating an information model derived from a body of source code, comprising: establishing a connection with a server computer; requesting an information model from the server, wherein the information model is derived from a particular body of source code; and receiving a visual presentation of the requested information model comprising a plurality of hierarchically arranged components and a plurality of documentation files.

Claim 77
A system **executing on a computer and stored on a computer readable-medium** for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: means for providing a visual presentation of the information model representing the software architecture; and means for modifying the software architecture.

Claim 108
A system **executing on a computer and stored on a computer readable-medium** for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: an information model viewer having a data dependency viewer capable of presenting the data dependencies between components of the information model and a functional dependency viewer capable of presenting the functional dependencies between components of the information model; a system architect having an architect designer capable of reorganizing the hierarchical component structure of the information model; an information model editor having a text interface capable of allowing editing of documentation associated with the information model and a file interface capable of receiving new or modified documentation files associated with the information model; and an information model builder having a text interface capable of allowing editing of source code files included with the body of source code and a file
interface capable of receiving new or modified source code files for inclusion with the body of source code.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. § 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1, 3 – 109 are rejected under 35 U.S.C. 102(b) as being anticipated by Rational Rose version 4.0 released November 1996. The documentation kit contains several manuals. Some of the manuals include the following:

Rational Rose Version 4.0 from Rational Software Corporation

Using Rational Rose 4.0 **(RAT-UR)**

Round-Trip Engineering with Rational Rose/C++ **(RAT-C++)**

UML Booch & OMT Quick Reference for Rational Rose 4.0 **(RAT-QR)**

Extensibility Guide **(RAT-EX)**

A Rational Approach to Software Development Using Rational Rose **(RAT-AP)**

Extensibility Reference Manual **(NOT-USED)**

Since, these manuals were contained in the same product documentation set these constitute a single reference for the purpose of rejection under 35 U.S.C. § 102

Knowledge of the Ordinary Artisan in the Art

5. One of Ordinary skill in the art prior to the time of invention should be familiar with the concepts of Object Technology. Furthermore, the implementation of object technology in a programming language such as C++ would also be known to one of very ordinary skill. The Examiner has made of record the text book, "The Object Primer The Application Developer's

Guide to Object Orientation", by Scott Amber from 1995. This text book covers inherent concepts and principles of object technology.

Claim 1
Rational Rose version 4.0 anticipates an information model representing a software architecture, comprising: a plurality of language independent format objects, each language independent format object representing a discrete component, wherein the components are structurally related into an information model according to a software architecture.
Examiner's Response
Rational Rose provides the ability to perform Object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The object model is language independent and is used to generate code (RAT-C++, Chapter 2 page 28 model to code correspondence) and Components (RAT-UR, Chapter 3, page 16 see Windows. Structurally related in an object model (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc...). Software architecture is inherent in object modeling and the limitation is met by the relationship of the modeling of objects with classes and component modeling and the generation of code from the models as indicated above.

Claim 3
The information model of claim 1, further comprising: one or more derivative language independent format objects, wherein a derivative language independent format object represents two or more combined language independent format objects; and a hierarchical structure relating the plurality of language independent format objects and the derivative language independent format objects into a software architecture.
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications. Class diagrams in object modeling are inherently hierarchical (RAT-UR, Chapter 4, Class modeling). (Components RAT-UR, pages 15 – 16, RAT-UR, Chapter 7)

Claim 4
The information model of claim 1, further comprising: one or more derivative view objects, wherein a derivative view object contains a structural relationship between two or more language independent format objects such that the one or more derivative view objects, in combination with the language independent format objects reflect a software architecture.
In view of claim 1 the principle of inheritance (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc...).

Claim 5
The information model of claim 1, wherein the software architecture is derived from a body of source code. (RAT-C++, pages 181 – 183 and 227)

Claim 6
Rational Rose version 4.0 anticipates a system for creating an information model representing an inherent software architecture derived from a body of source code, comprising: an information model generator having a parser and a composer, the parser configured to extract program fragments from a body of source code and create a plurality of language dependent format objects, the composer configured to convert language dependent format objects into language independent format objects, wherein each language independent format object represents a discrete component in an information model.
Examiner's Response
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 – 183 and 227) is used as input and parsed with a parser (RAT-C++, parsing, page 184) and the model is exported (RAT-C++, pages 185 – 186). The code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model)). Software architecture is inherent in object modeling and the components are structurally related.

Claim 7
The system of claim 6 wherein the composer is further configured to create a derivative language independent format object, the derivative language independent format object comprising two or more language independent format objects and representing a discrete component in the information model. Component diagram as per claim 3.

Claim 8
The system of claim 6 wherein the composer is further configured to analyze a file system structure of the body of source code and create a derivative view object, the derivative view object relating two or more language independent format objects into a discrete component in the information model based on the file system structure of the body of source code.
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 – 183 and 227) is used as input and parsed with a parser (RAT-C++, parsing, page 184) and the model is exported (RAT-C++, pages 185 – 186). The code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model – independent format with discrete component)). Software architecture is inherent in object modeling and the components are structurally related.

Claim 9
The system of claim 6 wherein the composer is further configured to analyze a configuration file and create a derivative view object, the derivative view object relating two or more language independent, format objects into a discrete component based on the configuration file. As per claim 8.

Claim 10
Rational Rose version 4.0 anticipates a method for creating an information model representing an inherent software architecture derived from a body of source code, comprising: extracting

program fragments from a body of source code; converting the program fragments to a language independent format; and creating a plurality of language independent format objects, wherein each language independent format object contains related program fragments from the body of source and represents a discrete component in an information model.
Examiner's Response
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 – 183 and 227) is used as input and parsed with a parser (RAT-C++, parsing, page 184) and the model is exported (RAT-C++, pages 185 – 186). The code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model – independent format with discrete component)). Software architecture is inherent in object modeling and the components are structurally related.

Claim 11
The method of claim 10, wherein the related program fragments are related by data dependencies, functional dependencies, and control flow indicators. (RAT-UR, diagrams Chapters 2 – 8, Class, Use Case, Interactions, Collaboration, Component and state and RAT-C++, Chapter 3, Reverse Engineering).

Claim 12
The method of claim 11, wherein the data dependencies comprise variables passed through function calls (Messaging is an inherent principle of object technology – RAT-UR, page 112 - 119) and shared global variables (RAT-UR, page 113).

Claim 13
The method of claim 11, wherein the functional dependencies comprise incoming function calls and outgoing function calls. As per claim 12.

Claim 14
The method of claim 10, further comprising the step of: creating one or more derivative language independent format objects, wherein a derivative language independent format object comprises two or more language independent format objects. As per claim 3.

Claim 15
The method of claim 1 o, further comprising the step of creating one or more derivative view objects, wherein a derivative view object relates two or more language independent format objects into a discrete component in the information model. As per claim 7.

Claim 16
The method of claim 1 5 , wherein the two or more language independent format objects are related based on the file system structure of the body of source code.
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 – 183 and 227) is used as input and parsed with a parser (RAT-C++, parsing, page 184) and the model is exported (RAT-C++, pages 185 – 186). The

code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model – independent format with discrete component)). Software architecture is inherent in object modeling and the components are structurally related.

Claim 17
The method of claim 15, wherein the two or more language independent format objects are related based on a configuration file. As per claim 16.

Claim 18
Rational Rose version 4.0 anticipates a method for creating an information model representing an inherent software architecture derived from a body of source code, comprising: parsing a body of source code to extract data dependencies, functional dependencies, and control flow indicators; creating a plurality of language dependent format objects, each language dependent format object comprising an abstract syntax tree representing related program fragments within the body of source code; and converting each language dependent format object into a language independent format object, wherein each language independent format object represents a discrete component in an information model.
Examiner's Response
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 - 183) is used as input and parsed with a parser (RAT-C++, parsing, page 184 – parsing inherently creates ASTs see 1956 article from Noam Chomsky) and the model is exported (RAT-C++, pages 185 – 186). The code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model – independent format with discrete component)). Components are representative of objects modeling which contain data dependencies, functional dependencies and control flow indicators (RAT-UR, diagrams Chapters 2 – 8, Class, Use Case, Interactions, Collaboration, Component and state). Software architecture is inherent in object modeling and the components are structurally related.

Claim 19
The method of claim 18, wherein the related program fragments are related by data dependencies, functional dependencies, and control flow indicators. As per claim 11.

Claim 20
The method of claim 19, wherein the data dependencies comprise variables passed through function calls and shared global variables. As per claim 12.

Claim 21
The method of claim 19, wherein the functional dependencies comprise incoming function calls and outgoing function calls. As per claim 12.

Claim 22

The method of claim 18, further comprising the step of creating one or more derivative language independent format objects, wherein a derivative language independent format object comprises two or more language independent format objects. As per claim 7.

Claim 23
The method of claim 18, further comprising the step of: creating one or more derivative view objects, wherein a derivative view object relates two or more language independent format objects into a discrete component in the information model. As per claim 3.

Claim 24
The method of claim 23, wherein the two or more language independent format objects are related based on the file system structure of the body of source code. As per claim 1 - Inherent in object modeling classes contain source code by definition a class contains attributes and the methods (source code) to perform operations on the attributes. RAT-C++, Chapter 3, Reverse Engineering,.

Claim 25
The method of claim 23, wherein the two or more language independent format objects are related based on a configuration file. (RAT-C++, Chapter 2, page 8)

Claim 26
Rational Rose version 4.0 anticipates a method for creating an information model representing a software architecture, comprising: creating a plurality of language independent format objects, wherein each language independent format object represents a discrete component in an information model.
Examiner's Response
Rational Rose provides the ability to perform Object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The object model is language independent and is used to generate code (RAT-C++, Chapter 2 page 28 model to code correspondence) and Components (RAT-UR, Chapter 3, page 16 see Windows. Structurally related in an object model (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc...). Software architecture is inherent in object modeling and the limitation is met by the relationship of the modeling of objects with classes and component modeling and the generation of code from the models as indicated above.

Claim 27
The method of claim 26, further comprising: arranging the plurality of language independent format objects into a hierarchical structure representing a software architecture.
The object model (inherently hierarchical) is language independent and is used to generate code (RAT-C++, Chapter 2 page 28 model to code correspondence) and Components (RAT-UR, Chapter 3, page 16 see Windows. Structurally related in an object model (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc...). Software architecture is inherent in object modeling and the limitation is met by the relationship

of the modeling of objects with classes and component modeling and the generation of code from the models

Claim 28
The method of claim 27, wherein the plurality of language independent format objects are arranged according to data dependencies, functional dependencies, and control flow indicators. As per claim 11.

Claim 29
The method of claim 28, wherein the data dependencies comprise variables passed through function calls and shared global variables. As per claim 12.

Claim 30
The method of claim 2,3, wherein the functional dependencies comprise incoming function calls and outgoing function calls. As per claim 12.

Claim 31
The method of claim 26, further comprising the step of: creating one or more derivative language independent format objects, wherein a derivative language independent format object comprises two or more language independent format objects. As per claims 1 and 3.

Claim 32
The method of claim 26. further comprising the step of creating one or more derivative view objects, wherein a derivative view object relates two or more language independent format objects into a discrete component in the information model. As per claims 1 and 3.

Claim 33
Rational Rose version 4.0 anticipates a system for manipulating an information model representing a software architecture, comprising: an information model viewer configured to provide a visual presentation of the information model representing the software architecture; and a system architect configured to modify the software architecture.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications.
The object model is language independent and is used to generate code (RAT-C++, Chapter 2 page 28 model to code correspondence) and Components (RAT-UR, Chapter 3, page 16 see Windows. Structurally related in an object model (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc...). The views Rational Rose provides are documented (RAT-UR, Chapter 3, pages 11 – 16 and Chapters 6, 7 and 8). Software architecture is inherent in object modeling and the limitation is met by the relationship of the modeling of objects with classes and component modeling and the generation of code from the models as indicated above.

Claim 34
The system of claim 33, wherein the information model viewer further comprises: a data dependency viewer configured to present the data dependencies between components of the information model; a functional dependency viewer configured to present the functional dependencies between components of the information model; and a calling tree viewer configured to present the control flow between program fragments contained within a component of the information model. Diagrams of claim 11.

Claim 35
The system of claim 34, :further comprising: a search results viewer configured to present the results of searches conducted within one or more information models or within one or more components of one or more information models. (RAT-UR, page 19, Browse different models).

Claim 36
The system of claim 33, further comprising: a language specific viewer configured to provide a visual presentation of the information model representing the software architecture according to one or more programming language paradigms. (RAT-UR, page 19, Browse different models and RAT-UR, page 200, UML, Basic or C++).

Claim 37
The system of claim 33;, wherein the system architect further comprises: an architect designer configured to reorganize the hierarchical component structure of the information model. (RAT-C++, page 59).

Claim 38
The system of claim 37, wherein the architect designer is further configured to merge two or more components together into a single component. RAT-UR, pages 121 - 122, inheritance provide with Dependency.

Claim 39
The system of claim 38, wherein the architect designer is further configured to fragment a merged component into two or more components. RAT-UR, pages 121 - 122, inheritance provide with Dependency (multiple).

Claim 40
The system of claim 37, wherein the system architect further comprises: an architect enhancer configured to add new components to the information model. RAT-UR, pages 121 – 122.

Claim 41
The system of claim 40, wherein the system architect further comprises: an architect creator configured to create new components and relate the newly created components into a new information model having no underlying body of source code. RAT-UR, page 122, Task Specification, Subprogram Specification, Package Specification.

Claim 42
The system of claim 41, wherein the system architect further comprises: an architect optimizer configured to extract functionally related components of an information model and create a new information model having a reduced set of components serving a desired function. RAT-C++, pages 59 - 61, Remove Selected, Remove All, and RAT-C++, pages 63 – 65, code regeneration.

Claim 43
The system of claim 33, further comprising: an information model editor having a text interface and a file interface, the text interface configured to allow editing of documentation associated with the information model and the file interface configured to receive new or modified documentation files associated with the information model. RAT-C++, pages 78 – 82.

Claim 44
The system of claim 33, further comprising: an information model builder having a text interface and a file interface, the text interface configured to allow editing of source code files included with the body of source code and the file interface configured to receive new or modified source code files for inclusion with the body of source code. RAT-C++, pages 180 – 183,

Claim 45
The system of claim 44, wherein the file interface is further configured to provide source code files from the body of source code. RAT-C++, pages 180 – 183,

Claim 46
The system of claim 33, further comprising: an information model search engine configured to accept a query, search the information model, and provide search results. RAT-UR, Chapter 12, pages 185 – 189.

Claim 47
The system of claim 33, further comprising: an information model document generator configured to compile a plurality of documentation objects into an information model document. RAT-C++, Reverse Engineering, Chapter 3,

Claim 48
The system of claim 33, further comprising: an information model difference generator configured to compare at least two information models and determine the differences between the at least two information models. RAT-C++, Chapter 4, Model Differencing

Claim 49
Rational Rose version 4.0 anticipates a method for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: merging two or more components into a compound component.
Examiner's Response

Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications. Class diagrams in object modeling are inherently hierarchical (RAT-UR, Chapter 4, Class modeling). The limitation of merging two or more components into a compound component (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation).

Claim 50
The method of claim 49, further comprising: modifying the hierarchical structure between two or more components; RAT-C++, page 113.

Claim 51
Rational Rose version 4.0 anticipates a method for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: fragmenting a compound component into its constituent components.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications. Class diagrams in object modeling are inherently hierarchical (RAT-UR, Chapter 4, Class modeling). The limitation of fragmenting a compound component into its constituent components (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation where the class is a Abstract class or mixin (Booch term from MIT) this is considered intended use of Rational Rose and knowledge within one of ordinary skill in the art).

Claim 52
The method of claim 51, further comprising: modifying the hierarchical structure between two or more components. RAT-C++, page 59.

Claim 53
Rational Rose version 4.0 anticipates a method for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: modifying the hierarchical structure between two or more components in an information model; merging two or more components into a compound component; and fragmenting a compound component into its constituent components.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications. Class diagrams in object modeling are inherently hierarchical (RAT-UR, Chapter 4, Class modeling). The limitation of merging two or more components into a compound component (is the inherent

principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation) and the limitation of fragmenting a compound component into its constituent components (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation where the class is a Abstract class or mixin (Booch term from MIT) this is considered intended use of Rational Rose and knowledge within one of ordinary skill in the art).

Claim 54
The method of claim 53, wherein the modifying step further comprises: selecting a first component; and promoting the first component relation to a second component. RAT-UR, Chapter 7, pages 122 - 123.

Claim 55
The method of claim 53, wherein the modifying step further comprises: selecting a first component; and demoting the first component in relation to a second component. RAT-UR, Chapter 7, pages 122 - 123.

Claim 56
The method of claim 53, wherein the merging step further comprises merging a component and a first compound component into a second compound component. RAT-UR, Chapter 7, pages 122, dependency.

Claim 57
The method of claim 53, wherein the merging step further comprises merging a first compound component and a second compound component into a third compound component. RAT-UR, Chapter 7, pages 122, dependency – intended use of the product - inheritance.

Claim 58
The method of claim 53, wherein the fragmenting step further comprises fragmenting a compound component into two or more components. RAT-UR, Chapter 7, pages 122, dependency – intended use of the product – inheritance – mixin as per claim 51.

Claim 59
The method of claim 53, wherein the fragmenting step further comprises fragmenting a compound component into a component and a compound component. RAT-UR, Chapter 7, pages 122, dependency – intended use of the product – inheritance.

Claim 60
The method of claim 53, wherein the fragmenting step further comprises fragmenting a compound component into a first compound component and a second compound component. As per claim 59.

Claim 61
The method of claim 53, further comprising editing documentation associated with the information model through a documentation text editor. A plurality of documentation files can be

interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2).

Claim 62
The method of claim 53, further comprising uploading new or modified documentation files through a file interface. A plurality of documentation files can be interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2).

Claim 63
The method of claim 53, further comprising compiling documentation associated with the information model into. an information model document. A plurality of documentation files can be interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2.

Claim 64
The method of claim 53, further comprising: creating a new component in the information model; and organizing the new component within the hierarchical structure of the information model. RAT-C++, page 113

Claim 65
The method of claim 53, further comprising searching the information model in response to a query and providing search results. (RAT-UR, page 19, Browse different models).

Claim 66
The method of claim 53, further comprising: comparing a first information model to a second information model; and generating a difference set containing the differences between the first information model and the second information model identified by the comparison. RAT-C++, Chapter 4, Model Differencing.

Claim 67
The method of claim 66, wherein the difference set comprises a new information model. RAT-C++, page 207.

Claim 68
Rational Rose version 4.0 anticipates a method for manipulating an information model derived from a body of source code, comprising: establishing a connection with a server computer; requesting an information model from the server, wherein the information model is derived from a particular body of source code; and receiving a visual presentation of the requested information model comprising a plurality of hierarchically arranged components and a plurality of documentation files.
Examiner's Response
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 – 183 and page 227) is used as input and parsed with a parser (RAT-C++, parsing, page 184) and the model is exported (RAT-C++, pages 185 – 186).

The code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model)). Software architecture is inherent in object modeling and the components are structurally related. RAT-UR, Chapter 13, Documentation Report. The limitation of a server with a connection is met by the support for Rose Configuration Management (CM) RAT-UR, Chapter 10. The documentation does not explicitly state the word *server* however on RAT-UR, page 157 the ability to establish the virtual path to the configuration management machine is shown and the well known functions of CheckOut, CheckIn and AcceptChanges are on RAT-UR, page 157. RAT-UR, page 174 mentions PVCS is supplied with the product. The ability to receive a visual presentation of the requested information model (RAT-UR, Chapter 3, page 16), class diagrams are inherently hierarchically arranged components (RAT-UR, page 16, Component window). A plurality of documentation files can be interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2).

Claim 69
The method of claim 68, wherein the plurality of components comprises: a plurality of single components; a plurality of compound components, wherein a compound component comprises two or more single components; and a plurality of data dependencies, functional dependencies, and control indicators, wherein data dependencies, functional dependencies, and control indicators relate the single components and compound components. As per claims 1 and 3.

Claim 70
The method of claim 69, wherein a compound component comprises one or more single components and one or more compound components. As per claims 1 and 3.

Claim 71
The method of claim 69, wherein a compound component comprises two or more compound components. As per claims 1 and 3.

Claim 72
The method of claim 68, further comprising: viewing a compound component and a sub-component of the compound component; selecting the sub-component; and viewing the data dependencies, functional dependencies, and control indicators of the sub-component. RAT-UR, Chapter 7, page 122.

Claim 73
The method of claim 68, further comprising: viewing a documentation file; and editing the documentation file. RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2.

Claim 74
The method of claim 68, further comprising: rearranging the hierarchical structure of the components. RAT-C++, page 113.

Claim 75
The method of claim 68, further comprising: submitting a search request for a particular component; receiving a search response, wherein the search response presents the requested component according to its relative position in the hierarchical structure. RAT-C++, page 113.

Claim 76
The method of claim 75, wherein the search response further presents each higher level component disposed between the requested component and a highest level component. (RAT-UR, page 19, Browse different models).

Claim 77
Rational Rose version 4.0 anticipates a system for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: means for providing a visual presentation of the information model representing the software architecture; and means for modifying the software architecture.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications.
The object model is language independent and is used to generate code (RAT-C++, Chapter 2 page 28 model to code correspondence) and Components (RAT-UR, Chapter 3, page 16 see Windows. Structurally related in an object model (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc...). The views Rational Rose provides are documented (RAT-UR, Chapter 3, pages 11 – 16 and Chapters 6, 7 and 8). Software architecture is inherent in object modeling and the limitation is met by the relationship of the modeling of objects with classes and component modeling and the generation of code from the models as indicated above.

Claim 78
The system of claim 77, wherein the information model viewer further comprises: means for presenting the data dependencies between components of the information model; means for presenting the functional dependencies between components of the information model; rind means for presenting the control flow between program fragments contained within a component of the information model. As per claim 77.

Claim 79
The system of claim 78, further comprising: means for presenting a result of a search conducted within one or more information models or within one or more components of one or more information models. (RAT-UR, page 19, Browse different models).

Claim 80
The system of claim 79, further comprising: means for presenting the information model as if the underlying body of source code was in a particular programming language. RAT-C++, Chapter 3, Reverse Engineering.

Claim 81
The system of claim 77, wherein the system architect further comprises: means for reorganizing the hierarchical component structure of the information model; means for merging two or more components together into a compound component; means for fragmenting a compound component into two or more components; and means for adding new components to the information model. As per claims 1 and 3.

Claim 82
The system of claim 81, wherein the system architect further comprises: means for creating new components; and means for relating the newly created components into a new information model having no underlying body of source code. RAT-UR, page 122, Task Specification, Subprogram Specification, Package Specification.

Claim 83
The system of claim 81, wherein the system architect further comprises: means for extracting functionally related components of an information model; and means for creating a new information model having a reduced set of components serving a desired function. RAT-C++, pages 185 – 187.

Claim 84
The system of claim 77. further comprising: means for editing documentation associated with the information model; and means for receiving new or modified documentation files associated with the information model. As per claim 83.

Claim 85
The system of claim 77, further comprising: means for editing source code files included with the body of source code; and means for receiving new or modified source code files for inclusion with the body of source code. As per claim 83.

Claim 86
The system of claim 85, wherein the means for receiving new or modified source code files further comprises: means to provide source code files from the body of source code. RAT-C++, Chapter 3, Reverse Engineering in view of claim 83.

Claim 87
The system of claim 77 , further comprising: means for accepting a query; means for searching the information model; and means for providing results. (RAT-UR, page 19, Browse different models).

Claim 88
The system of claim 77, further comprising: means for compiling a plurality of documentation objects into an information model document. A plurality of documentation files can be interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2.

Claim 89
The system of claim 77, further comprising: means for comparing at least two information models; and means for determining the differences between the at least two information models. A plurality of documentation files can be interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2.

Claim 90
Rational Rose version 4.0 anticipates a computer readable medium having stored thereon one or more sequences of instructions for causing one or more microprocessors to perform the steps for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, the steps comprising: merging two or more components into a compound component.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications.
The object model is language independent and is used to generate code (RAT-C++, Chapter 2 page 28 model to code correspondence) and Components (RAT-UR, Chapter 3, page 16 see Windows. Structurally related in an object model (RAT-UR, page 40, Diagram toolbar related through aggregation, Association, Link Attribute, Dependency etc…). The views Rational Rose provides are documented (RAT-UR, Chapter 3, pages 11 – 16 and Chapters 6, 7 and 8). Software architecture is inherent in object modeling and the limitation is met by the relationship of the modeling of objects with classes and component modeling and the generation of code from the models as indicated above. Emphasis on RAT-UR, pages 12 – 15, where the class diagram (inherently hierarchical) is used to formulate components which are show in the component diagram window through the use of inheritance (The limitation of merging two or more components into a compound component (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation – look over has-a and part-of relationships), this is considered intended use of Rational Rose and knowledge within one of ordinary skill in the art)..

Claim 91
The computer readable medium of claim 90 further comprising the step of: modifying the hierarchical structure between two or more components in an information model. RAT-C++, page 113.

Claim 92
Rational Rose version 4.0 anticipates a computer readable medium having stored thereon one or more sequences of instructions for causing one or more microprocessors to perform the steps for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, the steps comprising: fragmenting a compound component into its constituent components.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications. Class diagrams in object modeling are inherently hierarchical (RAT-UR, Chapter 4, Class modeling). The limitation of fragmenting a compound component into its constituent components (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation where the class is a Abstract class or mixin (Booch term from MIT) this is considered intended use of Rational Rose and knowledge within one of ordinary skill in the art).

Claim 93
The computer readable medium of claim 92 further comprising the step of: modifying the hierarchical structure between two or more components in an information model. RAT-C++, page 113.

Claim 94
Rational Rose version 4.0 anticipates a computer readable medium having stored thereon one or more sequences of instructions for causing one or more microprocessors to perform the steps for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, the steps comprising: modifying the hierarchical structure between two or more components in an information model; merging two or more components into a compound component; and fragmenting a compound component into its constituent components.
Examiner's Response
Rational Rose provides the ability to perform object modeling (information model) by diagramming object with a class diagram (RAT-UR, Chapter 4 – page 39 see figure). The ability to manipulate the object model (information model) is in Chapters 4 – 6 with specific mention of toolbars on pages 21 – 32 of Chapter 3 the Introduction to Diagrams and Specifications. Class diagrams in object modeling are inherently hierarchical (RAT-UR, Chapter 4, Class modeling). The limitation of merging two or more components into a compound component (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation) and the limitation of fragmenting a compound component into its constituent components (is the inherent principle of Inheritance in object technology) (RAT-UR, page 40, Association and Aggregation where the class is a Abstract class or mixin (Booch term from MIT) this is considered intended use of Rational Rose and knowledge within one of ordinary skill in the art).

Claim 95
The computer readable medium of claim 94, wherein the modifying step further comprises: selecting a first component; and promoting the first component in relation to a second component. As per claim 54.

Claim 96
The computer readable medium of claim 94, wherein the modifying step further comprises: selecting a first component; and demoting the first component in relation to a second component. As per claim 55.

Claim 97
The computer readable medium of claim 94, wherein the merging step further comprises merging a component and a first compound component into a second compound component. As per claim 56.

Claim 98
The computer readable medium of claim 94, wherein the merging step further comprises merging a first compound component and a second compound component into a third compound component. As per claim 57.

Claim 99
The computer readable ;medium of claim 94, wherein the fragmenting step further comprises fragmenting ,a compound component into two or more components. As per claim 58

Claim 100
The computer readable medium of claim 94, wherein the fragmenting step further comprises fragmenting a compound component into a component and a compound component. As per claim 57

Claim 101
The computer readable medium of claim 94, wherein the fragmenting step further comprises fragmenting a compound component into a first compound component and a second compound component. As per claim 57.

Claim 102
The computer readable medium of claim 94, further comprising editing documentation associated with the information model through a documentation text editor. A plurality of documentation files can be interpreted as both documentation such as RAT-UR, Chapter 13 and the generated files RAT-C++, Chapter 2

Claim 103
The computer readable medium of claim 94, further comprising uploading new or modified documentation files through a file interface. (RAT-C++, Chapter 3, Reverse Engineering, pages 180 – 181).

Claim 104
The computer readable medium of claim 94, further comprising compiling documentation associated with the information model into an information model document. (RAT-UR, Chapter 13, page 199 "NOTE:")

Claim 105
The computer readable medium of claim 94, further comprising: creating a new component in the information model; and organizing the new component within the hierarchical structure of the information model. RAT-C++, page 113.

Claim 106
The computer readable medium of claim 94, further comprising searching the information model in response to a query and providing search results. (RAT-UR, page 19, Browse different models).

Claim 107
The computer readable medium of claim 94, further comprising: comparing a first information model to a second information model; and generating a difference set containing the differences between the first information model and the second information model identified by the comparison. RAT-C++, Chapter 4, Model Differencing.

Claim 108
Rational Rose version 4.0 anticipates a system for manipulating an information model having a plurality of components arranged in a hierarchical structure representing a software architecture, comprising: an information model viewer having a data dependency viewer capable of presenting the data dependencies between components of the information model and a functional dependency viewer capable of presenting the functional dependencies between components of the information model; a system architect having an architect designer capable of reorganizing the hierarchical component structure of the information model; an information model editor having a text interface capable of allowing editing of documentation associated with the information model and a file interface capable of receiving new or modified documentation files associated with the information model; and an information model builder having a text interface capable of allowing editing of source code files included with the body of source code and a file interface capable of receiving new or modified source code files for inclusion with the body of source code.
Examiner's Response
Rational Rose version 4.0 supports reverse engineering (RAT-C++, Chapter 3) where a body of source code (RAT-C++, pages 181 – 183 and 227) is used as input and parsed with a parser (RAT-C++, parsing, page 184 – parsing inherently creates ASTs) and the model is exported (RAT-C++, pages 185 – 186). The code is converted into a model (Composer) (RAT-C++, page 185, second paragraph – generated model file and component package (informational model – independent format with discrete component)). A software architecture, comprising: an information model viewer having a data dependency viewer capable of presenting the data

dependencies between components of the information model (RAT-UR, page 98, collaboration diagram) and a functional dependency viewer capable of presenting the functional dependencies between components of the information model (RAT-UR, page 15 component diagram) ; a system architect having an architect designer capable of reorganizing the hierarchical component structure of the information model (RAT-UR, page 14, class diagram); an information model editor having a text interface capable of allowing editing of documentation associated with the information model (RAT-C++, pages 189 to 193) and a file interface capable of receiving new or modified documentation files associated with the information model (RAT-C++, pages 194 – 196); and an information model builder having a text interface capable of allowing editing of source code files included with the body of source code (RAT-C++, pages 194 – 196); and a file interface capable of receiving new or modified source code files for inclusion with the body of source code (RAT-C++, pages 194 – 196). Also note Documentation Report RAT-UR, Chapter 13).

Claim 109

The system of claim 108, further comprising: an information model viewer having a calling tree viewer capable of presenting the control flow between components of the information model and within a single component of the information model, and a language specific viewer capable of presenting the information model according to one or more programming language paradigms; a system architect having an architect enhancer capable of adding new components to the information model, an architect creator capable of creating new components and relating the newly created components into a new information model having no underlying body of source code, an architect optimizer capable of extracting functionally related components of an information model and creating a new information model having a reduced set of components serving a desired function; an information model search engine capable of accepting a query, searching the information model, and prodding search results; an information model document generator capable of compiling a plurality of documentation objects into an information model document; and an information model difference generator capable of comparing at least two information models, and determining differences between the at least two information models. The limitations of claim 109 are covered in the claims above.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rational Rose version 4.0 in view of **XML** Makes Object Models More Useful, Bruce Klein Information Week, June 28, 1999.

Claim 2
The information model of claim 1, wherein a language independent format object comprises: one or more XML files conforming to a document type definition describing a particular component; and one or more documentation objects containing information pertaining to the particular component.

Rational teaches the object model being language independent as per claim but Rational Rose from 1996 does not teach the use of XML. It is XML who teaches the use of XML with object models. Therefore, it would have been obvious to one of ordinary skill at the time of invention because, XML makes object models more useful.

Correspondence Information

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Todd Ingberg whose telephone number is (703) 305-9775.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703) 305-9662. Please, note that as of August 4, 2003 the FAX number changed for the organization where this application or proceeding is assigned is (703) 872-9306.

Also, be advised the United States Patent Office new address is

Post Office Box 1450

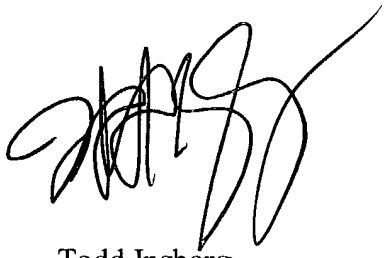
Alexandria, Virginia 22313-1450

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-9700.

Special Notice

9. Please, Note the Examiner's telephone number will change on October 23, 2004 when the Art Unit moves to the new location. The Examiner's new telephone number will be as follows:

(571) 272-3723



Todd Ingberg
Primary Examiner
Art Unit 2124
October 1, 2004